Intro to bsyncr

Cory Mosiman

June 9th, 2020

Overview

bsyncr is intended to be used in direct coordination with the nmecr package (even incorporating portions of nmecr into individual functions), enabling serialization of NMEC analysis into BuildingSync documents. bsyncr has methods to:

- Create generic, high level BSync documents
- Utilize nmecr generated models
- Serialize important information about the models, including:
    - Baseline and Reporting periods
    - Which variable is being predicted (Reporting Variable), its units, and enduse
    - Which variables are the covariates (Explanatory Variables) and their units
    - The parameters of the model generated from the baseline period (DerivedModelParameters)
    - The performance of the model generated from the baseline period (DerivedModelPerformance)
    - Other model outputs

Stub out a Generic BSync Document

In general, the following workflow can be performed:

1. Generate a root document
2. Stub out a building, optionally providing an id for the building
3. Stub out two scenarios, a baseline and reporting scenario

```
schema_loc <- "https://raw.githubusercontent.com/BuildingSync/schema/c620c7e58688698901edcb8560cd3e1b4b34d97
1/BuildingSync.xsd"
bsync_doc <- bsyncr::bs_gen_root_doc(schema_loc) %>%
  bsyncr::bs_stub_bldg(bldg_id = "My-Fav-Building") %>%
  bsyncr::bs_stub_scenarios(linked_building_id = "My-Fav-Building")
```

Stub out a Baseline and Reporting Scenario

For both the Baseline and Reporting Scenarios:

1. Grab the scenario
2. Stub out the necessary contents for the Derived Model

```
baseline_xpath <- "//auc:Scenario[auc:ScenarioType/auc:CurrentBuilding/auc:CalculationMethod/auc:Measured]"
reporting_xpath <- "//auc:Scenario[auc:ScenarioType/auc:PackageOfMeasures/auc:CalculationMethod/auc:Measured
]"

sc_baseline <- xml2::xml_find_first(bsync_doc, baseline_xpath)
not_used <- sc_baseline %>% bsyncr::bs_stub_derived_model(dm_id = "DerivedModel-Baseline",
                                           dm_period = "Baseline",
                                           sc_type = "Current Building")

sc_reporting <- xml2::xml_find_first(bsync_doc, reporting_xpath)
not_used <- sc_reporting %>% bsyncr::bs_stub_derived_model(dm_id = "DerivedModel-Reporting",
                                           dm_period = "Reporting",
                                           sc_type = "Package of Measures")
```

Generate nmecr model

Utilize the nmecr package to generate a simple linear regression (SLR) model.

```r
start_dt <- "03/01/2012 00:00"
end_dt <- "02/28/2013 23:59"
data_int <- "Daily"

b_df <- nmecr::create_dataframe(eload_data = nmecr::eload,
                                temp_data = nmecr::temp,
                                start_date = start_dt,
                                end_date = end_dt,
                                convert_to_data_interval = data_int)

SLR_model <- nmecr::model_with_SLR(b_df,
                                   nmecr::assign_model_inputs(regression_type = "SLR"))
```

Serialize the nmecr model using bsyncr

```r
dm_base_xpath <- "//auc:DerivedModel[auc:DerivedModelPeriod = 'Baseline']/auc:DerivedModelInputs"
dm_baseline <- xml2::xml_find_first(bsync_doc,
                                    dm_base_xpath)

not_used <- bs_gen_dm_nmecr(nmecr_baseline_model = SLR_model,
                            x = dm_baseline)
```

Write the file to output

```r
if (!dir.exists("output") ) {
  dir.create("output")
}
not_used <- xml2::write_xml(bsync_doc, "output/test1.xml")
```